Lecture - Implementation and numerical aspects of DG-FEM in 1D

Ph.D. Course:
Nodal DG-FEM for solving partial differential equations

Allan P. Engsig-Karup
Scientific Computing Section
DTU Informatics
Technical University of Denmark

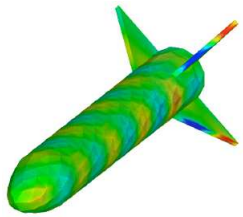
August 7, 2012

Course content

The following topics are covered in the course

1. Introduction & DG-FEM in one spatial dimension
2. Implementation and numerical aspects (1D)
3. Insight through theory
4. Nonlinear problems
5. Extensions to two spatial dimensions
6. Introduction to mesh generation
7. Higher-order operators
8. Problem with three spatial dimensions and other advanced topics

Requirements



What do we want?

- A flexible and generic framework useful for solving different problems
- Easy maintenance by a component-based setup
- Splitting of the mesh and solver problems for reusability
- Easy implementation

Let's see how this can be achieved...

Domain of interest

We want to solve a given problem in a domain Ω represented as a union of K nonoverlapping local elements D^k, $k = 1, ..., K$ such that

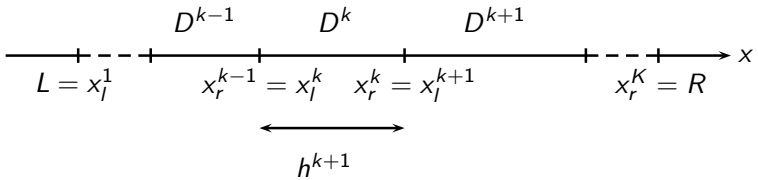
$$\Omega \cong \Omega_h = \bigcup_{k=1}^{K} D^k$$

Thus, we need to deal with implementation issues with respect to the local elements and how they are related.

The shape of the local elements can in principle be of any shape, however, in practice we mostly consider d-dimensional simplexes (e.g. triangles in two dimensions).

Sketch and notations for a one-dimensional domain

Consider a one-dimensional domain defined on $x \in [L, R]$



Local approximation in 1D

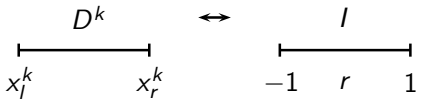
On each of the local elements, we choose to represent the solution locally as a polynomial of arbitrary order $N = N_p - 1$ as

$$x \in D^k : u_h^k(x(r), t) = \sum_{n=1}^{N_p} \hat{u}_n^k(t)\psi_n(r) = \sum_{i=1}^{N_p} u_i^k(t) l_i^k(r)$$

using either a modal or nodal representation on $r \in I([-1, 1])$.

We have introduced the affine mapping from the standard element I to the k'th element D^k

$$x \in D^k : x(r) = x_l^k + \frac{1+r}{2} h^k, \quad h^k = x_r^k - x_l^k$$



Local approximation in 1D

The modal representation can be based on a hierarchical modal basis, e.g. the normalized Legendre polynomials from the class of orthogonal Jacobi polynomials

$$\psi_n(r) = \tilde{P}_{n-1}(r) = \frac{P_{n-1}(r)}{\sqrt{\gamma_{n-1}}}, \quad \gamma_n = \frac{2}{2n+1}$$

which can be generated using a stable recurrence relation.

Note, the orthogonal property implies that for a normalized basis

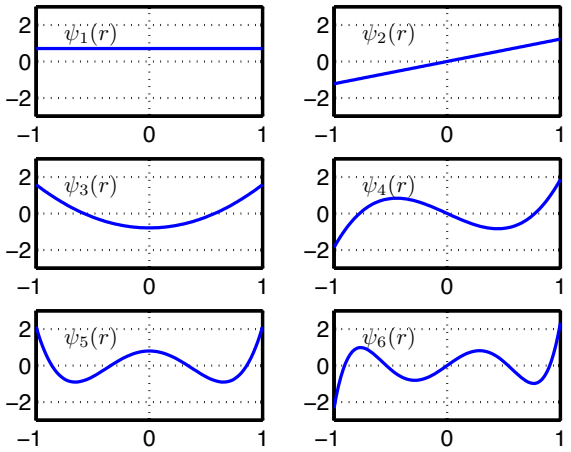
$$\int_{-1}^{1} \tilde{P}_i(r)\tilde{P}_j(r)dr = \delta_{ij}$$

The nodal representation is based on the interpolating Lagrange polynomials, which enjoys the Cardinal property

$$l_i^k(x) = \delta_{ij}, \quad \delta_{ij} = \begin{cases} 0 & , i \neq j \\ 1 & , i = j \end{cases}$$

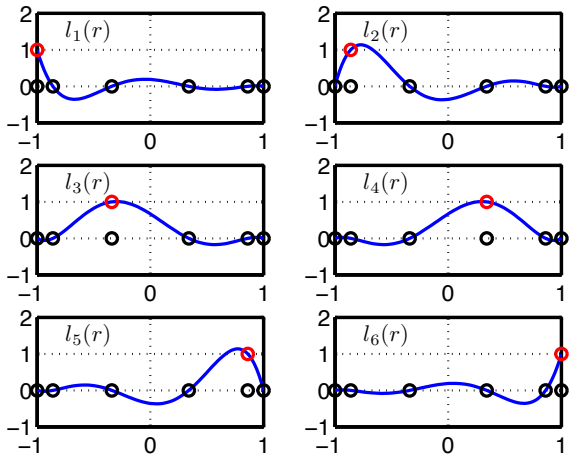
Local approximation in 1D - modes

Interpolating normalized Legendre polynomials $\psi_n(r) = \tilde{P}_{n-1}(r)$.



Local approximation in 1D - nodes

Interpolating Lagrange polynomials $l_n(r)$ based on the Legendre-Gauss-Lobatto (LGL) nodes.



Local approximation in 1D

The duality between using a modal or nodal interpolating polynomial representation is related through the choice of modal representation $\psi_n(r)$ and the distinct nodal interpolation points $\xi_i \in [-1, 1]$, $i = 1, ..., N_p$.

We can express this as

$$u(\xi_i) = \sum_{n=1}^{N_p} \hat{u}_n \psi_n(\xi_i) = \sum_{n=1}^{N_p} u_n l_n(\xi_i), \quad i = 1, ..., N_p$$

which defines a relationship between modal and nodal coefficients

$$\mathbf{u} = \mathcal{V}\hat{\mathbf{u}}$$

where

$$\mathcal{V}_{ij} = \psi_j(\xi_i), \quad \hat{\mathbf{u}}_\mathbf{i} = \hat{u}_i, \quad \mathbf{u}_\mathbf{i} = u(\xi_i)$$

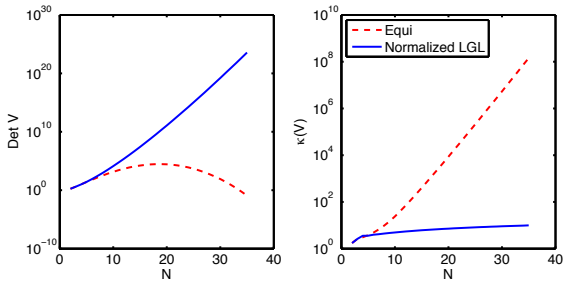
Local approximation in 1D

For stable and accurate computations we need to ensure that the generalized Vandermonde matrix $\mathcal{V}$ is well-conditioned. This implies minimizing the Lebesque constant

$$\Lambda = \max_r \sum_{i=1}^{N_p} |l_i(r)|, \quad l_i(r) = \prod_{j=0, j\neq i}^{N_p} \frac{r - \xi_j}{\xi_i - \xi_j}$$

$$||u - u_h||_\infty \leq (1 + \Lambda)||u - u^*||_\infty$$

and maximizing Det $\mathcal{V}$ through choice of node distribution.

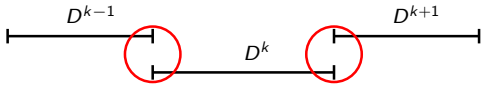


Global approximation

The global solution $u(x, t)$ can then be approximated by the direct summation of the local elemental solutions

$$u_h(x, t) = \bigoplus_{k=1}^{K} u_h^k(x, t)$$

Recall, at the traces of adjacent elements there will be two distinct solutions. Thus, there is an ambiguity in terms of representing the solution.



Approximation of the PDE

As an example, consider the PDE for a general conservation law in 1D space

$$\partial_t u + \partial_x f(u) = 0, \quad x \in [L, R]$$

For the approximation of the unknown solution $u(x,t)$ we make use of the expansion $u_h(x,t)$ and insert it into the PDE we want to solve.

Doing this, result in an expression for the residual $\mathcal{R}_h(x,t)$. We require that the test functions are orthogonal to the Residual function

$$\mathcal{R}_h(x,t) = \partial_t u_h + \partial_x f(u_h)$$

in the Galerkin sense as

$$\int_{D^k} \mathcal{R}_h(x,t)\psi_n(x)dx = 0, \quad 1 \leq n \leq N_p$$

on each of the $k = 1, ..., K$ elements.

Local approximation

As we have seen, in a DG-FEM discretization we can apply a polynomial expansion basis of arbitrary order within each element.

Thus, to exploit this in a code we need procedures for

- computing polynomial expansions

$$u(\xi_i) = \sum_{n=1}^{N_p} \hat{u}_n \psi_n(\xi_i) = \sum_{n=1}^{N_p} u_n l_n(\xi_i), \quad i = 1, ..., N_p$$

- numerical evaluation of integrals and derivatives

$$\mathcal{M}^k \frac{du_h^k}{dt} + \mathcal{S} f_h^k - \mathcal{M}^k g_h^k = (f_h^k - f^*)\delta_{1j} - (f_h^k - f^*)\delta_{N_p j}$$

Element-wise operations

To implement the algorithms we need a way to determine the element-wise operators.

Consider the mass matrix $\mathcal{M}^k$ for the k'th element defined as

$$\mathcal{M}_{ij}^k = \int_{x_l^k}^{x_r^k} l_i^k(x) l_j^k(x) dx = \frac{h^k}{2} \int_{-1}^{1} l_i(r) l_j(r) dr = \frac{h^k}{2} \mathcal{M}_{ij}$$

with $\mathcal{M}$ the standard mass matrix.

Consider the stiffness matrix $\mathcal{S}^k$ for the k'th element defined as

$$\mathcal{S}_{ij}^k = \int_{x_l^k}^{x_r^k} l_i^k(x) \frac{dl_j^k(x)}{dx} dx = \int_{-1}^{1} l_i(r) \frac{dl_j(r)}{dr} dr = \mathcal{S}_{ij}$$

with $\mathcal{S}$ the equivalent standard stiffness matrix.

Local matrices that are time-invariant can be pre-calculated and stored for reuse.

Element-wise operations

The integrals in the elements of the local matrix operators can be determined using high-order accurate Gaussian quadrature rules of the form

$$\int_{-1}^{1} u(\xi)(1-\xi)^\alpha (1+\xi)^\beta d\xi = \sum_{n=0}^{N} u(\xi_n) w_n + \mathcal{R}(u)$$

which can exactly integrate polynomials $u(\xi) \in \mathcal{P}^{2N+2-k}$ if it is based on Gauss points ($k = 1$), e.g. the Gauss-Legendre quadrature ($\alpha = \beta = 0$)

```
>> [x,w]=JacobiGQ(alpha,beta,N)
```

suitable for the types of integrals in the local operators of the 1D DG-FEM formulation.

However, there is an alternative and computationally convenient way to determine the needed local operators exactly...

Element-wise operations

Earlier, a relationship between the coefficients of our modal and nodal expansions was found

$$\mathbf{u} = \mathcal{V}\hat{\mathbf{u}}$$

Also, we can relate the modal and nodal bases through

$$\mathbf{u}^T\mathbf{l}(r) = \hat{\mathbf{u}}^T\psi(r)$$

These relationships can be combined to relate the basis functions through $\mathcal{V}$

$$\begin{aligned}
&\mathbf{u}^T\mathbf{l}(r) = \hat{\mathbf{u}}^T\psi(r) \\
\Rightarrow &\left(\mathcal{V}\hat{\mathbf{u}}\right)^T\mathbf{l}(r) = \hat{\mathbf{u}}^T\psi(r) \\
\Rightarrow &\hat{\mathbf{u}}^T\mathcal{V}^T\mathbf{l}(r) = \hat{\mathbf{u}}^T\psi(r) \\
\Rightarrow &\mathcal{V}^T\mathbf{l}(r) = \psi(r)
\end{aligned}$$

Element-wise operations

We have just found the compact relationship

$$\mathcal{V}^T\mathbf{l}(r) = \psi(r)$$

which can be expressed directly as

$$\begin{bmatrix} \psi_1(\xi_1) & \psi_1(\xi_2) & \cdots & \psi_1(\xi_{N_p}) \\ \psi_2(\xi_1) & \psi_2(\xi_2) & \cdots & \psi_2(\xi_{N_p}) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_{N_p}(\xi_1) & \psi_{N_p}(\xi_2) & \cdots & \psi_{N_p}(\xi_{N_p}) \end{bmatrix} \begin{bmatrix} l_1(r) \\ l_2(r) \\ \vdots \\ l_{N_p}(r) \end{bmatrix} = \begin{bmatrix} \psi_1(r) \\ \psi_2(r) \\ \vdots \\ \psi_{N_p}(r) \end{bmatrix}$$

It turns out to be useful for determining the standard elemental matrices...

Element-wise operations

The i'th Lagrange polynomial can be expressed as

$$l_i(r) = \sum_{n=1}^{N_p} \left(\mathcal{V}^T\right)^{-1}_{in} \psi_n(r)$$

Insert into an element of the standard mass matrix

$$\begin{aligned}
\mathcal{M}_{ij} &= \int_{-1}^{1} \sum_{n=1}^{N_p} \left(\mathcal{V}^T\right)^{-1}_{in} \psi_n(r) \sum_{m=1}^{N_p} \left(\mathcal{V}^T\right)^{-1}_{jm} \psi_m(r) dr \\
&= \sum_{n=1}^{N_p}\sum_{m=1}^{N_p} \left(\mathcal{V}^T\right)^{-1}_{in} \left(\mathcal{V}^T\right)^{-1}_{jm} (\psi_n, \psi_m)_I \\
&= \sum_{n=1}^{N_p} \left(\mathcal{V}^T\right)^{-1}_{in} \left(\mathcal{V}^T\right)^{-1}_{jn}
\end{aligned}$$

which is equivalent to

$$\mathcal{M} = \left(\mathcal{V}\mathcal{V}^T\right)^{-1} \quad \Rightarrow \quad \mathcal{M}^k = \frac{h^k}{2}\mathcal{M}$$

Element-wise operations I

We choose to express the derivative of the i'th Lagrange polynomial as

$$\frac{dl_j(r)}{dr} = \sum_{n=1}^{N_p} \frac{dl_j(r)}{dr}\Big|_{r_n} l_n(r)$$

Now, consider an element of the stiffness matrix

$$\begin{aligned}
\mathcal{S}_{ij} &= \int_{-1}^{1} l_i(r) \sum_{n=1}^{N_p} \frac{dl_j(r)}{dr}\Big|_{r_n} l_n(r) dr \\
&= \sum_{n=1}^{N_p} \frac{dl_j(r)}{dr}\Big|_{r_n} \int_{-1}^{1} l_i(r) l_n(r) dr
\end{aligned}$$

Element-wise operations II

Introduce the differentiation matrix

$$\mathcal{D}_{r,(i,j)} = \left.\frac{dl_j}{dr}\right|_{r_i}$$

Then, we find that

$$\mathcal{S} = \mathcal{M}\mathcal{D}_r$$

The entries of the differentiation matrix is found directly from

$$\mathcal{V}^T\mathcal{D}_r^T = \mathcal{V}_r^T, \quad \mathcal{V}_{r,(i,j)} = \left.\frac{d\psi_j}{dr}\right|_{r_i}$$

Thus, we can compute the differentiation matrix from

$$\mathcal{D}_r = (\mathcal{V}^T)^{-1}(\mathcal{V}_r)^T = \mathcal{V}_r\mathcal{V}^{-1}$$

Element-wise operations

The classical orthogonal Jacobi polynomials $P_n^{(\alpha,\beta)}(r)$ of order n can be selected as the local modal basis functions $\psi_n(r)$.

The orto-normalized Jacobi polynomials form the following weighted inner product

$$\int_{-1}^{1} \tilde{P}_i^{(\alpha,\beta)}(r)\tilde{P}_j^{(\alpha,\beta)}(r)w(r)dr = \delta_{ij}, \quad r \in [-1,1]$$

where the weight function is $w(r) = (1-r)^\alpha(1+r)^\beta$.

Furthermore, an important property relates the derivatives of one class of polynomials to the functions of another class as

$$\frac{d}{dx}\tilde{P}_n^{(\alpha,\beta)}(r) = \sqrt{n(n+\alpha+\beta+1)}\tilde{P}_n^{(\alpha+1,\beta+1)}(r)$$

which can be exploited to determine the described local operators in a straightforward and computationally convenient way.

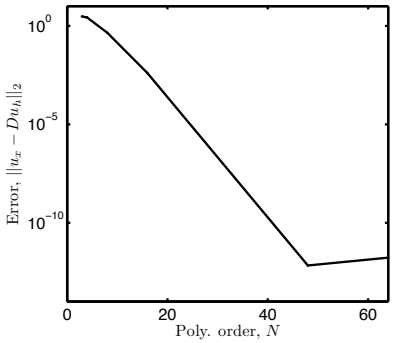
Element-wise operations

A summary of useful scripts for the 1D element operations in Matlab

JacobiP	Evaluate normalized Jacobi polynomial
GradJacobiP	Evaluate derivative of Jacobi polynomial
JacobiGQ	Compute Gauss points and weights for use in quadrature
JacobiGL	Compute the Legende-Gauss-Lobatto (LGL) nodes
Vandermonde1D	Compute $\mathcal{V}$
GradVandermonde1D	Compute $\mathcal{V}_r$
Dmatrix1D	Compute $\mathcal{D}_r$

Spectral accuracy

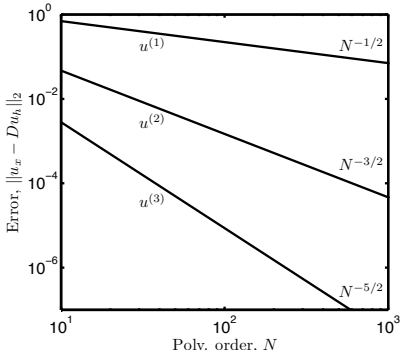
$$u(x) = \exp(\sin(\pi x)), \quad x \in [-1,1]$$



$$||e_h||_2 = ||u_x - D_r u_h||_2 = \sqrt{e_h^T\mathcal{M}e_h}$$

Smoothness of functions and numerical accuracy

$$u^{(0)}(x) = \begin{cases} -\cos(\pi x) & , -1 \leq x < 0 \\ \cos(\pi x) & , 0 \leq x \leq 1 \end{cases}, \quad \frac{du^{(i+1)}}{dx} = u^{(i)}, \quad i = 0, 1, 2, ...$$



Note: only algebraic convergence observed since $u^{(i+1)} \in C^i$.

$$||\frac{du^{(i)}}{dx} - \mathcal{D}_r u_h^{(i)}||_{\Omega_h} \propto N^{1/2-i}$$

A second look at DG-FEM

The basics of DG-FEM

Let us consider the proto-type problem

$$u_t + f_x = 0, \quad x \in \Omega([L, R]), \quad f(u) = au$$

By using the Energy Method we find the energy estimate

$$\frac{d}{dt}||u||_\Omega^2 = -a(u^2(R) - u^2(L))$$

Which suggest the need for imposing the following BCs in finite domains

$$u(L, t) = g(t), \quad \text{if } a \geq 0$$
$$u(R, t) = g(t), \quad \text{if } a \leq 0$$

Energy conservation when problem has a periodic nature

$$u(R) = u(L)$$

With this in mind, let's consider the DG-FEM scheme...

The basics of DG-FEM

Let us consider the strong form of DG-FEM for the same problem

$$\mathcal{M}^k \frac{d}{dt} u_h^k + \mathcal{S}(au_h^k) = [I^k(x)(au_h^k - f^*)]_{x_l^k}^{x_r^k}$$

For the analysis of the scheme, we note that

$$u_h^T \mathcal{M}^k u_h = \int_{D^k} u_h^2 dx = ||u_h^k||_{D^k}^2$$
$$u_h^T S u_h = \int_{D^k} u_h^k(x) \frac{d}{dx} u_h^k(x) dx = \tfrac{1}{2}[(u_h^k)^2]_{x_l^k}^{x_r^k}$$

Thus, we can derive the following discrete local energy estimate

$$\frac{d}{dt}||u_h^k||_{D^k}^2 = -a[(u_h^k)^2]_{x_l^k}^{x_r^k} + 2[u_h^k(au_h^k - f^*)]_{x_l^k}^{x_r^k}$$

Contruct a discrete global energy estimate and require

$$\frac{d}{dt}||u_h||_{\Omega,h}^2 = \sum_{k=1}^{K} \frac{d}{dt}||u_h^k||_{D^k}^2 \leq 0$$

If this is fullfilled the scheme should be stable...

The basics of DG-FEM

From the discrete local energy estimate

$$\frac{d}{dt}||u_h^k||_{D^k}^2 = [a(u_h^k)^2 - 2u_h^k(au)^*]_{x_l^k}^{x_r^k}$$

we conclude that stability for our scheme must be controlled by

- choice of numerical flux, $f^* = (au)^*$
- handling of outer boundary conditions (imposed weakly through f^*)

Both of these choices should be motivated by the physical nature of the problem.

On a term-by-term we can try to ensure that the energy is non-increasing...

Standard notation for elements

It is customary to refer to the interior information of the element by a superscript "-" and the exterior by a "+".

Furthermore, this notation is used to define the average operator

$$\{\{u\}\} \equiv \frac{u^- + u^+}{2}$$

where u can be both a scalar and a vector.

Jumps can be defined along an outward point normal $\hat{\mathbf{n}}$ (to the element in question) through the jump operator

$$[[u]] \equiv \hat{\mathbf{n}}^- u^- + \hat{\mathbf{n}}^+ u^+$$
$$[[\mathbf{u}]] \equiv \hat{\mathbf{n}}^- \cdot \mathbf{u}^- + \hat{\mathbf{n}}^+ \cdot \mathbf{u}^+$$

The basics of DG-FEM

We have the freedom to choose a numerical flux to guarantee the stability requirement. For example, consider

$$f^* = (au)^* = \{\{au\}\} + |a|\frac{1-\alpha}{2}[[u]]$$

- $\alpha = 0$: upwind flux

$$\begin{aligned} f^* = (au)^* &= \{\{au\}\} + |a|\frac{1-\alpha}{2}[[u]] \\ &= \tfrac{1}{2}a(u^- + u^+) + |a|\tfrac{1}{2}(\hat{\mathbf{n}}^- u^- + \hat{\mathbf{n}}^+ u^+) \\ &= \tfrac{1}{2}(a + |a|\hat{\mathbf{n}}^-)u^- + \tfrac{1}{2}(a + |a|\hat{\mathbf{n}}^+)u^+ \end{aligned}$$

- $\alpha = 1$: central/average flux

$$f^* = (au)^* = \{\{au\}\}$$

The basics of DG-FEM

For each internal interface (to the domain) we get a contribution from left and right

$$u_h(x^-)\hat{\mathbf{n}}^- \cdot (f(x^-) - 2f^*(x^-) + u_h(x^+)\hat{\mathbf{n}}^+ \cdot (f(x^+) - 2f^*(x^+))$$

By collecting these contributions from each interface to the energy integral, it is possible to show that we obtain interface contributions of the form

$$-|a|(1-\alpha)[[u_h^k]]^2$$

which is less than or equal to zero (≤ 0) for $0 \leq \alpha \leq 1$.

- when $\alpha \neq 1$, the role of the term is to introduce some dissipation whenever there is a jump on the interface.

The basics of DG-FEM

We assume the case

$$a > 0 \quad : \quad u(x_L, t) = g_-(t)$$

and we come up with two ways to handle the outer BCs

- **Approach #1:** (direct)

$$f_L = a g_-(t), \quad f_R = a u_h^K(x_r^K)$$

- **Approach #2:** (symmetry)

$$f_L = -a u_h^1(x_l^1) + 2a g_-(t), \quad f_R = a u_h^K(x_r^K)$$

Now, assume that $g(t) = 0$ to mimic that no external source can change the energy of the system.

The basics of DG-FEM

- Approach #1: (direct)

$$\frac{d}{dt}||u_h||^2_{\Omega,h} = \underbrace{-|a|(1-\alpha)\sum_{k=1}^{K-1}[[u_h^k(x_r^k)]]^2}_{\text{Interior interfaces}} \underbrace{-(1-\alpha)|a|(u_h^1(x_l^1))^2}_{\text{Left BC}} \underbrace{-a(u_h^K(x_r^K))^2}_{\text{Right BC}}$$

- Approach #2: (symmetry)

$$\frac{d}{dt}||u_h||^2_{\Omega,h} = \underbrace{-|a|(1-\alpha)\sum_{k=1}^{K-1}[[u_h^k(x_r^k)]]^2}_{\text{Interior interfaces}} \underbrace{-a(u_h^1(x_l^1))^2}_{\text{Left BC}} \underbrace{-a(u_h^K(x_r^K))^2}_{\text{Right BC}}$$

Conclusion: The conditions for a stable scheme are

$$a \geq 0, \quad 0 \leq \alpha \leq 1 \quad \Rightarrow \quad \text{Stability!}$$

- Energy dissipation for $\alpha \neq 1$
- Energy conservation for $\alpha = 1$ and periodic condition $u(x_L) = u(x_R)$ (no BC terms then)

The basics of DG-FEM

What did we learn from this?

- Stability is enforced through the local flux choice(s)
- The numerical solutions are discontinuous between elements
- Boundary and interface conditions are imposed weakly
- All operators are local
- Due to the weak interface-based coupling, there are no restrictions on element size and local approximation
  - choices are to provide accuracy

These properties are what contributes to DG-FEM being a very flexible method

Putting the pieces together in a code

Putting the pieces together in a code

Consider the linear advection equation

$$\partial_t u + a\partial_x u = 0, \quad x \in [0, L]$$

with IC and BC conditions

$$u(x,0) = \sin\left(\tfrac{2\pi}{L}x\right), \quad u(0,t) = -\sin\left(\tfrac{2\pi}{L}t\right)$$

The exact solution to this problem is given as

$$u(x,t) = \sin\left(\tfrac{2\pi}{L}(x-at)\right)$$

Putting the pieces together in a code

The DG-FEM method for solving the linear advection equation on the k'th element is derived from the residual equation

$$\int_{\Omega^k} l_i^k(x)\partial_t u_h^k dx + a\int_{\Omega^k} l_i^k(x)\partial_x u_h^k dx = 0$$

where the local solution is represented as

$$u_h^k(x,t) = \sum_{i=1}^{N_p} u_h^k(x_i^k,t) l_i^k(x)$$

By integration by parts twice and exchanging the numerical flux

$$\int_{\Omega^k} l_i^k(x)\partial_t u_h^k dx + a\int_{\Omega^k} l_i^k(x)\partial_x u_h^k dx = \oint_{\partial\Omega^k} l^k \hat{n}\cdot(f_h^k - f_h^*)dx$$

where $i = 1, ..., N_p$.

Putting the pieces together in a code

By inserting the local approximation for the solution we can now obtain the local semidiscrete scheme

$$\mathcal{M}^k \frac{du_h^k}{dt} + a\mathcal{S}u_h^k = \delta_{N_p j}(f_h^k(x_r^k) - f^*) - \delta_{1j}(f_h^k(x_l^k) - f^*)$$

Then, we need to pick a suitable numerical flux f^* for the problem, e.g. the Lax-Friedrichs-type flux

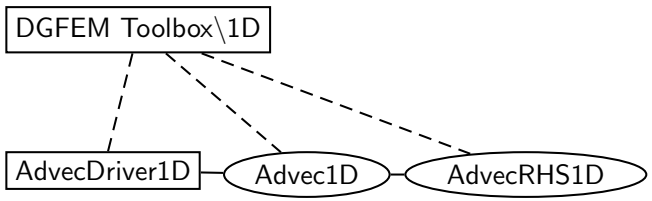
$$f^* = \{\{cu\}\} + a\frac{1-\alpha}{2}[[u]], \quad 0 \le \alpha \le 1$$

which we have already shown leads to a stable scheme.

Putting the pieces together in a code

To build your own solver using the DGFEM codes

AdvecDriver1D	Matlab main function for solving the 1D advection equation.
Advec1D(u,FinalTime)	Matlab function for time integration of the semidiscrete PDE.
AdvecRHS1D(u,time,a)	Matlab function defining right hand side of semidiscrete PDE



- Several examples, functions, utilities in DGFEM package
- Fast proto-typing
- Write your own solver independent on mesh

Element-wise operations

A summary of scripts needed for preprocessing for 1D DG-FEM computations in Matlab

Globals1D	Define list of globals variables
MeshGen1D	Generates a simple equidistant grid with K elements
Startup1D	Main script for pre-processing
BuildMaps1D	Automatically create index maps from conn. and bc tables
Normals1D	Compute outward pointing normals at elements faces
Connect1D	Build global connectivity arrays for 1D grid
GeometricFactors1D	Compute metrics of local mappings
Lift1D	Compute surface integral term in DG formulation

- Components listed here are general and reusable for each solver - enables fast proto-typing.
- MeshGen1D can be substituted with your own preferred mesh interface/generator.

Putting the pieces together in a code

```
% Driver script for solving the 1D advection equations
Globals1D;

% Order of polymomials used for approximation
N = 8;

% Generate simple mesh
[Nv, VX, K, EToV] = MeshGen1D(0.0,2.0,10);

% Initialize solver and construct grid and metric
StartUp1D;

% Set initial conditions
u = sin(x);

% advection speed
a = 2*pi;

% numerical flux (stable: 0<=alpha<=1)
alpha = 1;

% CFL constant
CFL=0.75;

% Solve Problem
FinalTime = 10;
[u] = Advec1D(u,FinalTime,a,alpha);
```


Putting the pieces together in a code

To solve a semidiscrete problem of the form

$$\frac{du_h}{dt} = \mathcal{L}_h(u_h, t)$$

employ some appropriate ODE solver to deal with time, e.g. the low-storage explicit fourth order Runge-Kutta method (LSERK4)[1]

$$\mathbf{p}^{(0)} = \mathbf{u}^n$$

$$i \in [1, ..., 5] : \begin{cases} \mathbf{k}^{(i)} = a_i \mathbf{k}^{(i-1)} + \Delta t \mathcal{L}_h(\mathbf{p}^{(i-1)}, t^n + c_i \Delta t) \\ \mathbf{p}^{(i)} = \mathbf{p}^{(i-1)} + b_i \mathbf{k}^{(i)} \end{cases}$$

$$\mathbf{u}_h^{n+1} = \mathbf{p}^{(5)}$$

- For every element, the time step size Δt has to obey a CFL condition of the form

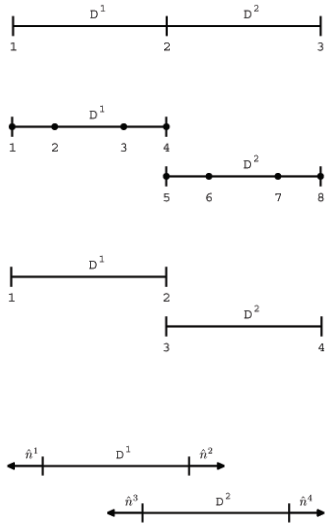
$$\Delta t \leq \frac{C}{a} \min_{k,i} \Delta x_i^k$$

[1] See book p. 64.

Putting the pieces together in a code

```
function [u] = Advec1D(u, FinalTime, a, alpha)
% Purpose  : Integrate 1D advection until FinalTime starting with
%            initial the condition, u

Globals1D;
time = 0;

% Runge-Kutta residual storage
resu = zeros(Np,K);

% compute time step size
dxmin = min(abs(x(1,:)-x(2,:)));
dt = CFL*dxmin/a;
Nsteps = ceil(FinalTime/dt); dt = FinalTime/Nsteps;

% outer time step loop
for tstep=1:Nsteps
    for INTRK = 1:5
        timelocal = time + rk4c(INTRK)*dt;
        [rhsu] = AdvecRHS1D(u, timelocal, a, alpha);
        resu = rk4a(INTRK)*resu + dt*rhsu;
        u = u+rk4b(INTRK)*resu;
    end;
    % Increment time
    time = time+dt;
end;
return
```

Putting the pieces together in a code



Mesh tables:
EToV = [1 2; 2 3];
VX = [0 0.5 1.0];

Global node numbering defined:
vmapM = [1 4 5 8];
vmapP = [1 5 4 8];

Face node numbering defined:
mapM = [1 2 3 4];
mapP = [1 3 2 4];
mapI = [1];
mapO = [4];

Outward point face normals:
nx = [-1 1; -1 1];

Putting the pieces together in a code

$$\frac{du_h^k}{dt} = -a(\mathcal{M}^k)^{-1}\mathcal{S}u_h^k + (\mathcal{M}^k)^{-1}\delta_{N_p j}(f_h^k(x_r^k) - f^*) - (\mathcal{M}^k)^{-1}\delta_{1j}(f_h^k(x_l^k) - f^*)$$

$$f^* = \{\{au\}\} + a\frac{1-\alpha}{2}[[u]], \quad 0 \le \alpha \le 1$$

Note: all interfaces treated same way in a single line of code.

```
function [rhsu] = AdvecRHS1D(u,time, a,alpha)

% function [rhsu] = AdvecRHS1D(u,time)
% Purpose  : Evaluate RHS flux in 1D advection

Globals1D;

% form flux differences at faces
df    = zeros(Nfp*Nfaces,K);
df(:) = 0.5*a*(u(vmapM)-u(vmapP)).*(nx(:)-(1-alpha));

% impose boundary condition at x=0
uin      = -sin(a*time);
df(mapI) = 0.5*a*(u(vmapI)-uin).*(nx(mapI)-(1-alpha));
df(mapO) = 0;

% compute right hand sides of the semi-discrete PDE
rhsu = -a*rx.*(Dr*u) + LIFT*(Fscale.*(df));
return
```


1D Advection equation

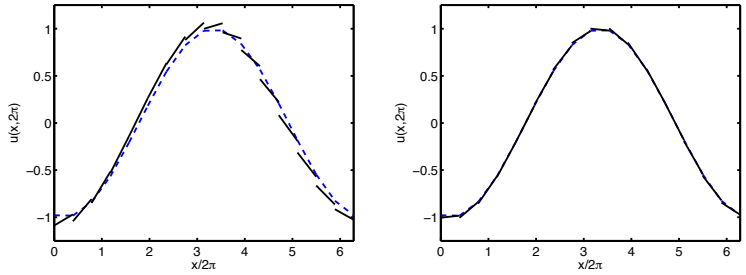


Figure: Computations in a finite domain. a) Central flux and b) Upwind flux. No. elements $K = 16$ and poly. order $N = 1$.

Examples: error behavior

Consider the simple advection equation on a periodic domain

$$\partial_t u - 2\pi\partial_x u = 0, \quad x \in [0, 2\pi], \quad u(x,0) = \sin(lx), \quad l = \frac{2\pi}{\lambda}$$

Exact solution is then $u(x,t) = \sin(l(x - 2\pi t))$.

Errors at final time $T = \pi$.

N\ K	2	4	8	16	32	64	Convergence rate
1	-	4.0E-01	9.1E-02	2.3E-02	5.7E-03	1.4E-03	2.0
2	2.0E-01	4.3E-02	6.3E-03	8.0E-04	1.0E-04	1.3E-05	3.0
4	3.3E-03	3.1E-04	9.9E-06	3.2E-07	1.0E-08	3.3E-10	5.0
8	2.1E-07	2.5E-09	4.8E-12	2.2E-13	5.0E-13	6.6E-13	$\cong$ 9.0

Error is seen to behave as

$$||u - u_h||_{\Omega,h} \le Ch^{N+1}$$

Examples: error behavior

What about time dependence?

Final time (T)	π	10π	100π	1000π	2000π
(N,K)=(2,4)	4.3E-02	7.8E-02	5.6E-01	>1	>1
(N,K)=(4,2)	3.3E-03	4.4E-03	2.8E-02	2.6E-01	4.8E-01
(N,K)=(4,4)	3.1E-04	3.3E-04	3.4E-04	7.7E-04	1.4E-03

Error is seen to behave as

$$||u - u_h||_{\Omega,h} \leq C(T)h^{N+1} \cong (c_1 + c_2 T)h^{N+1}$$

Examples: error behavior

What about cost?

N\K	2	4	8	16	32	64
1	1.00	2.19	3.50	8.13	**19.6**	54.3
2	2.00	3.75	**7.31**	15.3	38.4	110.
4	**4.88**	8.94	20.0	45.0	115.	327.
8	15.1	32.0	68.3	163.	665.	1271.
16	57.8	121.	279.	664.	1958.	5256.

Time$\cong C(T)K(N+1)^2$

N\K	2	4	8	16	32	64	Convergence rate
1	-	4.0E-01	9.1E-02	2.3E-02	**5.7E-03**	1.4E-03	2.0
2	2.0E-01	4.3E-02	**6.3E-03**	8.0E-04	1.0E-04	1.3E-05	3.0
4	**3.3E-03**	3.1E-04	9.9E-06	3.2E-07	1.0E-08	3.3E-10	5.0
8	2.1E-07	2.5E-09	4.8E-12	2.2E-13	5.0E-13	6.6E-13	$\cong$ 9.0

Higher order is cheaper

A few remarks

We know now

- It is the flux that gives stability
- It is the local basis that gives accuracy
- The scheme is very (VERY) flexible(!!!)

BUT - we have doubled the number of degrees of freedom along the interfaces.

In 1D not a big deal – penalty is $\frac{N+1}{N}$, however in multi-dimensional spaces this is something we need to be concerned about.

Does it generalize?

Let us first consider the scalar conservation law

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0, \quad x \in [L, R]$$

and rewrite into quasi-linear form

$$\frac{\partial u}{\partial t} + f_u(u)\frac{\partial u}{\partial x} = 0, \quad x \in [L, R]$$

Then in analogy with the linear advection equation the boundary conditions need to be according to

$$u(L,t) = g_1(t) \quad \text{when } f_u(u(L,t)) \geq 0$$
$$u(R,t) = g_2(t) \quad \text{when } f_u(u(R,t)) \leq 0$$

Assume as usual that

$$x \in D^k : u_h^k(x,t) = \sum_{n=1}^{N_p} \hat{u}_n^k(t)\psi_n(x) = \sum_{i=1}^{N_p} u_h^k(x_i^k, t)l_i^k(x)$$

Does it generalize?

From the conservation form we directly recover the weak form

$$\int_{D^k} \left(\frac{\partial u_h^k}{\partial t} \phi_j^k - f_h^k(u_h^k) \frac{d\phi_j^k}{dx} \right) dx = - \int_{\partial D^k} \hat{\mathbf{n}} \cdot f^* \phi_j^k dx$$

and the corresponding strong form

$$\int_{D^k} \left(\frac{\partial u_h^k}{\partial t} + \frac{\partial f_h^k(u_h^k)}{\partial x} \right) \phi_j^k dx = \int_{\partial D^k} \hat{\mathbf{n}} \cdot (f_h^k(u_h^k) - f^*) \psi_j^k dx$$

By multiplying with each of the test functions $\phi_j^k \in V_h$, $j = 1, ..., N_p$ yields exactly N_p equations for the local N_p unknowns.

Here a polynomial representation of the flux function has been introduced

$$x \in D^k : f_h(u_h^k) = \sum_{n=1}^{N_p} \hat{f}_n^k \psi_n(x) = \sum_{i=1}^{N_p} f_h(x_i^k) l_i^k$$

In 1D, the edge integral term is straightforward to evaluate.

Does it generalize?

The only thing that remains unknown is the flux

$$f^* = f^*(u_h^-, u_h^+)$$

We rely on the hugely succesfull theory of finite volume monotone schemes

- The numerical flux is consistent, $f^*(u_h, u_h) = f(u_h)$
- The numerical flux is monotone

$$f^*(a, b) = f^*(\uparrow, \downarrow)$$

Does it generalize?

There are many choices for the numerical flux to choose from

- Lax-Friedrichs flux

$$f^{LF}(a, b) = \frac{f(a) + f(b)}{2} + \frac{C}{2} \hat{\mathbf{n}} \cdot (a - b)$$

  - where the global LF flux is given by

$$C \geq \max_{\inf u_h(x) \leq s \leq \sup u_h(x)} |f_u(s)|$$

  - and the local LF flux is obtained by

$$C \geq \max_{\min(a,b) \leq s \leq \max(a,b)} |f_u(s)|$$

Does it generalize?

... but the FV literature is filled with alternatives

- Exact Riemann solvers
- Gudonov fluxes
- Engquist-Osher fluxes
- Approximate Riemann fluxes (Roe, Van Leer, HLLC, etc.)

Which choice is right is essentially determined by

- the problem physics

To keep things simple we shall mainly focus on the LF flux which generally works very well, but is also the most dissipative flux.

Does it generalize?

Let us now consider the system of conservation laws

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{f}(\mathbf{u})}{\partial x} = 0, \quad x \in [L, R]$$
$$\mathbf{u} = [u_1(x,t), ..., u_m(x,t)]^T$$

where the inflow boundary conditions are determined by the eigenvalues of the jacobian for the system

$$\mathcal{B}_L u(L,t) = \mathbf{g}_1(t) \quad \text{at } x = L$$
$$\mathcal{B}_R u(L,t) = \mathbf{g}_2(t) \quad \text{at } x = R$$

The only essential difference is that C in the LF flux depends on the eigenvalues of the jacobian for the system

$$C = \max_{\mathbf{u}} \left| \lambda \left(\frac{\partial \mathbf{f}}{\partial \mathbf{u}} \right) \right|$$

Does it generalize?

For multidimensional problems of the form

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathbf{f}(\mathbf{u}) = 0$$

with initial and inflow boundary conditions imposed. There is essentially no difference. The weak form

$$\int_{D^k} \left(\frac{\partial \mathbf{u}_h^k}{\partial t} \phi_i^k - \mathbf{f}_h^k(\mathbf{u}_h^k) \cdot \nabla \phi_i^k \right) d\mathbf{x} = - \int_{\partial D^k} \hat{\mathbf{n}} \cdot \mathbf{f}^* \phi_i^k d\mathbf{x}$$

The strong form

$$\int_{D^k} \left(\frac{\partial \mathbf{u}_h^k}{\partial t} + \nabla \cdot \mathbf{f}_h^k(\mathbf{u}_h^k) \right) \phi_i^k d\mathbf{x} = \int_{\partial D^k} \hat{\mathbf{n}} \cdot (\mathbf{f}_h^k(\mathbf{u}_h^k) - \mathbf{f}^*) \phi_i^k d\mathbf{x}$$

with the LF-flux

$$\mathbf{f}^* = \{\{\mathbf{f}_h(\mathbf{u}_h)\}\} + \frac{C}{2}[[\mathbf{u}_h]], \quad C = \max_u \left| \lambda \left(\hat{\mathbf{n}} \cdot \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \right) \right|$$

Boundary integrals over edges/faces in 2D/3D requires representation of flux function.

Summary

We already know a lot about the basic DG-FEM

- Stability is provided by carefully choosing the numerical flux.
- Accuracy appear to be given by the choice of local solution representation.
- Flexibility both in terms of geometry and range of problems that can be solved.
- We can utilize major advances on monotone schemes to design fluxes.
- The DG-FEM scheme generalizes with very few changes to very general problems – particularly it is well-suited for multidimensional systems of conservation laws.

At least in principle – but what can we actually prove?